EPJ Web of Conferences **251**, 04019 (2021) https://doi.org/10.1051/epjconf/202125104019
CHEP 2021

The Controls and Configuration Software of the ATLAS Data Acquisition System: evolution towards LHC Run 3

Andrei Kazarov[1,*], *Adrian* Chitan[5], *Andrei* Kazymov[2], *Alina* Corso-Radu[3], *Igor* Aleksandrov[2], *Igor* Soloviev[3], *Giuseppe* Avolio[4], *Matei* Vasile[5], and *Mikhail* Mineev[2]

[1]NRC "Kurchatov Institute" - PNPI, St. Petersburg, Russian Federation
[2]JINR, Dubna, Russian Federation
[3]University of California, Irvine, USA
[4]CERN, Geneva, Switzerland
[5]National Institute for Physics and Nuclear Engineering, Bukharest, Romania

Abstract. The ATLAS experiment at the Large Hadron Collider (LHC) operated very successfully in the years 2008 to 2018, in two periods identified as Run 1 and Run 2. ATLAS achieved an overall data-taking efficiency of 94%, largely constrained by the irreducible dead-time introduced to accommodate the limitations of the detector read-out electronics. Out of the 6% dead-time only about 15% could be attributed to the central trigger and DAQ system, and out of these, a negligible fraction was due to the Control and Configuration subsystem. Despite these achievements, and in order to improve even more the already excellent efficiency of the whole DAQ system in the coming Run 3, a new campaign of software updates was launched for the second long LHC shutdown (LS2). This paper presents, using a few selected examples, how the work was approached and which new technologies were introduced into the ATLAS Control and Configuration software. Despite these being specific to this system, many solutions can be considered and adapted to different distributed DAQ systems.

1 Introduction

The ATLAS experiment [1] at the Large Hadron Collider at CERN relies on a complex Trigger and Data Acquisition (TDAQ) system [2] to gather and select particle collision data at unprecedented energy and rate.

The TDAQ system is an overly complex distributed computing system composed of a large number of hardware and software components (about 3000 computers and more than 50000 concurrent processes) which, in a coordinated manner, provide the data-taking functionality of the overall system.

The Control and Configuration (CC) system [3] is responsible for the software infrastructure which manages and operates the various components of the system and integrates them with the wider ATLAS data taking environment.

It is the software component taking care of configuring, controlling and monitoring all the TDAQ components and it has to guarantee the smooth and synchronous operations of all the various sub-systems.

*e-mail: Andrei.Kazarov@cern.ch

The CC software ranges from high level applications to low level packages and it is designed following a layered component model (Figure 1). At the very bottom are external packages to deal, for instance, with threads. Next layer includes base libraries like in-house developed object persistency system OKS and the libraries for the CORBA-based interprocess communication. Higher up are a set of services, like the configuration service or the process management. Above these layers is the so-called application layer, with the run control, the expert system, and a set of graphical user interfaces (GUIs) allowing the operator to act on the system.

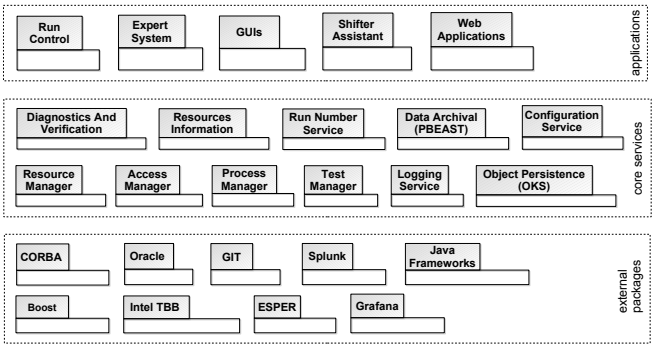

Figure 1. Control and Configuration software: the high-level architecture.

The CC system is a crucial actor in operating the TDAQ system. Indeed, a disruption in any of the basic and fundamental provided services would not only prevent to acquire LHC collisions data but would also undermine the capability to properly control and monitor a data taking session. For that reason, the CC system is asked to provide the means to minimize the downtime of the system caused by runtime failures.

2 CC Software evolution and upgrade towards LHC Run 3

Control and Configuration services and applications played an important role in successful TDAQ operations during data taking period in LHC Runs 1 and 2, allowing a high level of the data taking efficiency.

The first long LHC shutdown (LS1, from February 2013 to spring 2015) was primarily used to carry out a complete revision of the control and configuration software. Indeed, several packages were designed in late ’90s and developed in the next decade. Additionally, new requirements, not foreseen when the system was originally designed, emerged during the ATLAS operations and were implemented in a less than optimal way. At the same time, new software technologies appeared that could easily replace or simplify several custom-made solutions.

As a consequence, the goals for the LS1 updates were three-fold: properly accommodate additional requirements that could not be seamlessly included during steady operation of the system; re-factor software that had been repeatedly modified to include new features, thus becoming less maintainable; and seize the opportunity of modernizing the software, thus profiting from the rapid evolution in IT technologies. The LS1 CC updates are discussed in detail in [4].

Based on what was done during LS1 and the additional experience gained during Run 2, a new campaign of software updates was launched for the second long LHC shutdown (LS2).

LS2 provided a novel opportunity for significant improvements of various parts of the system and to make use of new technologies and standards.

All the upgrades were carried out retaining the important constraint of minimally impacting the mode of operation of the system and public APIs, in order to maximize the acceptance of the changes by the large user community.

3 Developments for Run 3

The following sections give a description of the strategic choices and approaches have been adopted for the development of some of the major components of the CC system.

3.1 Configuration service storage improvements

The service provides the data taking configuration parameters. Such data describe more than 50,000 online processes distributed on more than 3,000 nodes with details of their control, monitoring, diagnostic, recovery, data-flow and data quality configurations, as well as connectivity and parameters for various TDAQ and detector modules, chips and channels. The configurations are prepared and updated by many experts from various DAQ, high-level trigger and detector groups [5]. Their consistency is one of the key requirements for reliable and effective functionality of the TDAQ system and the whole ATLAS experiment.

The configuration service is based on the OKS database storing data in XML files, developed in the middle of the '90s [6] and gradually improved by needs of the experiment. During LHC Run 2 we performed an evaluation of available database technologies looking for suitable candidates for configuration data storage and distribution. It was decided to keep the OKS database for implementation with few changes.

3.1.1 Data format changes

The first change was an improvement of the OKS XML data format to make it slimmer and more readable by humans. In particular, now the data payload is stored into XML attributes instead of elements, empty values and any internal counting numbers are avoided. Thus, there is much less probability that an expert updating some XML file in a text editor can make a syntax mistake.

3.1.2 New repository storage: Git backend

The configuration is stored in more than 1000 XML files, with various experts responsible for updating the ones associated with their systems and detectors. In 2008, a special service was implemented to validate any changes before they are committed into OKS repository verifying the consistency of the files and the update permissions based on user roles [7]. The service was successfully used during LHC runs 1 and 2 keeping configuration data consistent 100% of time, however some changes are needed for the future.

The service was based on CVS [8], that has been used by the ATLAS software during its development. CVS isn't supported anymore for many years and misses important security and interface improvements. Nowadays, the Git [9] is an obvious replacement for the service implementation demonstrating many benefits and profiting expertise and commitment by our software experts. We avoided several issues of the CVS implementation keeping the service design mostly unchanged:

- Since Git uses transactions, now, several changes in interconnected files can be committed in one go. This was not the case in the CVS implementation, where every file was committed individually and a failure might leave the repository in an inconsistent state.
- In the CVS implementation a configuration was read from a repository snapshot on a shared file system updated by the CVS server after every commit. Thus, the processes for the same run might get different configurations depending of the moment they access it. In the new implementation a run configuration is preserved by a unique Git commit hash, thus any process reads the same configuration independently of ongoing commits.
- The old service implemented the repository validation on the client side and provided a special utility to commit changes on the CVS server, so the CVS interface was not exposed to the users. In the Git implementation the validation is performed in the Git server pre-receive hook, so clients can use any Git interface including web editors.
- When a configuration needs to be reloaded in the course of the data-taking session, the CVS implementation presented to an operator a list of modified files to be selected from for a new configuration. This error prone approach was replaced by selection of a configuration from newly available revisions with meaningful commit logs. The postponed changes are committed into git branches and git merge requests are used to handle them.

The OKS Git update workflow is presented in Figure 2. Gitea [10] is used as a Git server and it is only accessible inside the ATLAS experiment area. To modify a configuration, the user clones the repository, makes necessary changes, commits them and pushes back to gitea. The gitea hook validates changes, and, in case of success, stores them on the server, updates the read-only snapshot on NFS [11] and synchronizes with CERN GitLab server [12]. The NFS snapshot is only used for fast viewing of the configuration. The copy of repository on CERN GitLab can be used to read configuration outside of the experiment area and is accessible world-wide for authorized ATLAS users.

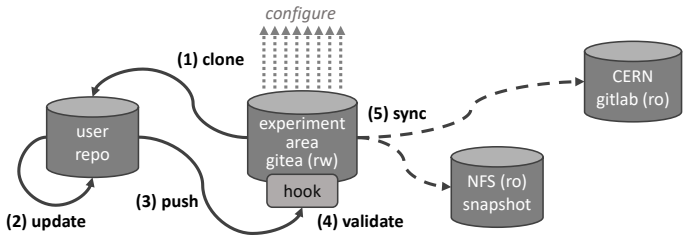

Figure 2. The OKS git update workflow

3.1.3 Archival

The last change is the new configuration archiving approach. In the past, every OKS configuration used for ATLAS data-taking run was archived in Oracle with fine grained details of individual attribute values of configuration objects. Thus, every new value of every attribute resulted in one or more new rows in the Oracle tables to be accessible independently of others. This was requested by the configuration task force before Run 1, but never used in practice since then. Instead, the archived configurations were accessed on the files granularity level only. It was then decided to drop the Oracle archiving and to use the CERN GitLab repository instead. Every OKS configuration revision can be accessed world-wide via secure

HTTPS protocol using CERN on CERN Single-Sign-On (SSO). Any configuration used for data-taking runs is tagged in git with the run number, and such tag is stored into the Run Number database. Thus, every archived OKS configuration can be easily accessed by its run number.

3.2 CHIP: expert system

CHIP (Central Hint and Information Processor) [13] was introduced for the first time in the TDAQ system during the LS1 period. CHIP is actually an automation and error management service whose task is two-fold: maximizing the system efficiency and optimizing the person power. Indeed, through a continuous and comprehensive monitoring of the TDAQ system, CHIP has proven in Run 2 to be able to deal fast and effectively with error and failures, thus greatly reducing the need of manual interventions by the operator. CHIP's tasks can be divided into 3 main categories:

- Handling abnormal conditions;
- Automating complex procedures;
- Performing advanced recoveries.

CHIP at its core relies on an open-source Java-based Complex Event Processing (CEP) [14] engine, ESPER [15]. The key features of the ESPER engine are the support for advanced stream analysis (correlation, aggregation, sliding windows, temporal patterns), a rich SQL-like Event Processing Language (EPL) to express the knowledge base, a natively high-configurable multi-threaded architecture, the support for historical data replication and built-in advanced metrics.
During Run 2, the performance of a single instance of CHIP[1] was adequate to monitor the whole TDAQ system and handle all the streams of information generated during the data taking sessions (Figure 3), with data injection peaks into the ESPER engine of about 40 kHz. The evaluation of EPL statements was very efficient too, with an average execution time of about 2 μs per statement (weighted by the number of executions of a statement).

For the LS2 updates, the knowledge base was extended even further to cover new scenarios and implement stop-less operations for new components of the TDAQ system (i.e., the new read-out [16]). Currently the knowledge base counts more than 340 EPL statements in 29 different contexts, corresponding to a 13% increase in number of statements with respect to Run 2.
Furthermore, LS2 was a good occasion to update the underlining ESPER engine to its last version 8. Even though the migration required some code modification and adaptation in CHIP, it brought sensible improvements in terms of performance and knowledge base organization. Indeed, the EPL statements are now compiled into Java byte code, hence improving the efficiency of the runtime execution, allowing a more fine-grained verification process and exposing features usually only available in high-level object-based programming language (e.g., the possibility to define streams of events as private in the context of a knowledge base module, thus greatly reducing the probability of hiding or overriding a certain data stream definition). As an example, the average execution time (wall time) for the most executed EPL statement could be reduced by more than 40%.

[1] Running on server equipped with two Intel Xeon E5-2680 V3 CPUs and 64 GB of RAM.

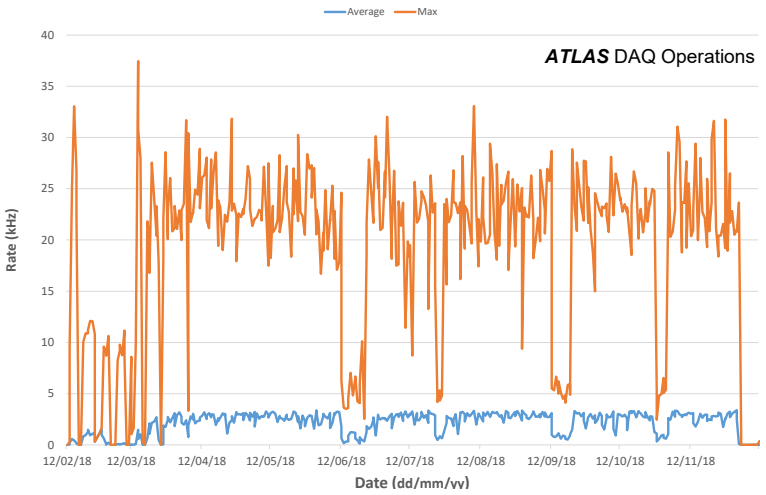

Figure 3. The average (blu line) and maximum (orange line) event injection rate into the ESPER engine over the whole 2018 ATLAS data taking period. As expected, the maximum injection rate has a high degree of variation. Indeed, the number of events generated by the DAQ system greatly depends on the state of the system itself (i.e., state transition, errors and anomalies). At the same time, the average rate is rather constant, with dips corresponding to the LHC machine development and maintenance periods.

3.3 WebRC: a web-based DAQ Run Control

3.3.1 Web applications and new requirements

In last decades, web applications are used more and more widely in the area of control and monitoring of the ATLAS experiment, gradually replacing traditional GUI applications (e.g. Qt-based). During LS2 we decided to develop a web application, providing functionality similar to the main controlling application for TDAQ data taking, called Integrated GUI (IGUI) [17] (Java-based). This functionality must include: connecting to a running TDAQ session for control or monitoring; presenting a hierarchical tree of different types of TDAQ applications, dynamically updating changes in their states; sending commands to controller applications; subscribing and browsing of Error Reporting Service (ERS) messages in real time; and less important things like application log files browsing. The application was named Web Run Control or WebRC.

3.3.2 Choice of technology: Apache Wicket

The main factors affecting the choice of technology for the WebRC application were the following:

- Its backend part needs to be tightly integrated with main TDAQ services like Run Control, Information Service and ERS;
- The frontend part should offer rich set of widgets, allowing to implement features similar to Java Swing elements;
- It shall be well scalable and conservative in resource usage, allowing connections for many users and serving multiple TDAQ partitions in parallel;
- Support of dynamic and interactive web features like Ajax or Web Sockets.

After a survey and a prototype development, a solid candidate was identified: Apache Wicket [18], an open source, component oriented, pure-Java web application framework. Developed since 2004, it remains one of the mainstream Java server-side frameworks, allowing to develop a powerful Java server application which integrates naturally into TDAQ software ecosystem. In addition, it does not bring additional libraries or requirements on the frontend side.

3.3.3 WebRC design and implementation

The server side of WebRC is a single multi-threaded Java application which provides HTTP access to Wicket components, modelling different elements of running DAQ partitions. The frontend part is a simple HTML page, containing visualization of the components, following a typical Model-View design pattern. Visualization is managed internally by Wicket, leaving to the programmer only the backend part of the business. The screenshot of the page with an example of a running partition is presented in Figure 4.

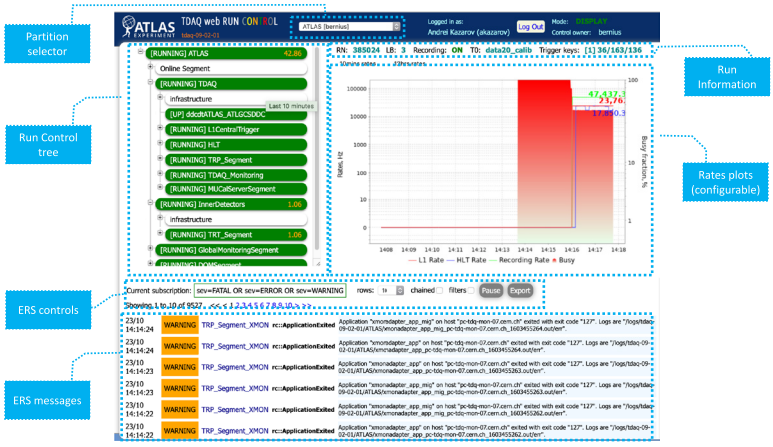

Figure 4. A screenshot of WebRC web page, representing a TDAQ partition in the Running state.

WebRC application does not require any additional infrastructure and it's single instance can serve dozens of clients and connections. It may be functioning in two different modes: first is DISPLAY mode, or monitoring mode, where no RC commands can be sent to the system, and second is CONTROL mode which provides full functionality like changing system states, restarting applications etc. This type of application facilitates a lot the remote access to the control and monitoring aspects of TDAQ data taking for system experts and operators, which is becoming very demanding in periods of teleworking. Implemented functionality allows to fully control an ATLAS data-taking session from a web browser, and potentially it may replace the traditional IGUI application for this purpose. It is possible and is foreseen to integrate the application with authentication services (like CERN Single-Sign-On) and with standard TDAQ authorisation facilities (Access Manager).

3.4 ELisA: an electronic logbook for ATLAS and more

A web facility for an electronic logbook [19] has been developed to keep track of the daily activities of the ATLAS operations, commissioning and deployment work. The logbook is

used by the operators, experts and automated services to record and share information. The logbook comprises a web user interface (as seen in Figure 5), a REST API, a set of client libraries, and a set of command line utilities for programmatic-free access to the logbook operations [20]. The logbook uses a database backend to store its configuration. The facility provides a configurable email notification mechanism. Also, a logbook message can be replied directly from the user's preferred mail client, without accessing the web interface. The logbook implemements restricted access with multiple user authentication mechanisms. Developed primarily as the ATLAS experiment operations logbook, the ELisA facility is cur-

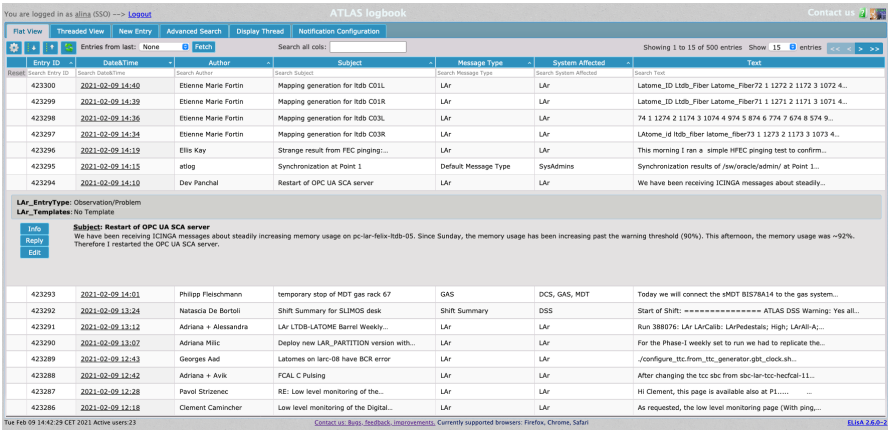

Figure 5. ELisA web interface screenshot.

rently being used for other projects such as detector development and commissioning work. During the LS2 period, we have mainly implemented solutions [21] to improve the portability of the logbook in private setups outside of the ATLAS working environment:

- We have added support for other database technologies (MYSQL) in order to avoid the dependency from the centralized Oracle database which is quite a complex operation to setup;
- We have extended the authentication mechanisms: beside the existing choices based on CERN Single-Sign-On (SSO) and LDAP servers, social media login (Google and Github) is now available. This could be potentially useful for setups outside CERN environment.
- We have reduced the development and maintenance effort by using Spring Boot framework. This framework eases the dependencies management, the XML configuration is rulled out almost entirely, an application is automatically configured given the project dependencies. As well, Spring Boot provides an embedded servlet container, thus eliminating the need for an external servlet container installation and maintenance work.
- We have improved the deployment procedure by packing all the necessary software and configuration into an RPM, which is available in a public repository.

4 Conclusions and Outlook

The Control and Configuration software has contributed to the physics results obtained by the ATLAS experiment during Run 1 by ensuring smooth and efficient data taking. It was completely revised during the Long Shutdown 1 (2013-2014) period in order to accommodate

additional requirements, improve maintainability and profit from advances in IT technologies. All this was done applying minimal changes to APIs, such that the large amount of client code would not need significant adaptations. The Control and Configuration software has proved to be stable, reliable and well performing in LHC Run 2 (2015-2018). In order to face the new challenges that will arise in Run 3 operations, the Control and Configuration software has undergone a further modernization process in different components during the Long Shutdown 2. The experience operating the TDAQ system, has also demonstrated that the overall modular architecture of the control and configuration system is flexible and supports partial upgrades, as well as step-wise modernization of its components. This is fundamental for a system that is foreseen to run for several more years and that will undergo several more upgrade iterations.

References

[1] The ATLAS Collaboration (ATLAS), JINST **3**, S08003 (2008)
[2] M. Abolins et al. (ATLAS TDAQ), JINST **11**, P06008 (2016)
[3] G. Lehmann Miotto et al., Nucl. Instrum. Meth. **A623**, 549 (2010)
[4] A. Kazarov, I. Aleksandrov, G. Avolio, M. Caprini, A. Chitan, A.C. Radu, A. Kazymov, G.L. Miotto, M. Mineev, A. Santos et al., Journal of Physics: Conference Series **1525**, 012036 (2020)
[5] I. Soloviev et al., Journal of Physics: Conference Series **119**, 022004 (2008)
[6] R. Jones, L. Mapelli, Y. Ryabov, I. Soloviev, IEEE Transactions on Nuclear Science **45**, 1958 (1998)
[7] I. Soloviev, Journal of Physics: Conference Series **396**, 012047 (2012)
[8] *CVS - Concurrent Versions System*, http://cvs.nongnu.org/, accessed: 2021-01-27
[9] *GIT*, https://git-scm.com/, accessed: 2021-01-27
[10] *Gitea - git with a cup of tea*, https://gitea.io/, accessed: 2021-01-27
[11] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, *Design and implementation or the sun network filesystem* (1985)
[12] *GitLab - The complete DevOps platform*, https://gitlab.com/, accessed: 2021-01-27
[13] G. Anders, G. Avolio, G.L. Miotto, L. Magnoni, J. Phys. Conf. Ser. **608**, 012007 (2015)
[14] *Complex Event Processing*, https://en.wikipedia.org/wiki/Complex_event_processing, accessed: 2019-04-29
[15] *ESPER*, http://www.espertech.com, accessed: 2019-04-29
[16] The ATLAS Collaboration, J. Phys. Conf. Ser. **898**, 032057 (2017)
[17] G. Avolio, M. Caprini, G.L. Miotto, Journal of Physics: Conference Series **331**, 022032 (2011)
[18] *Apache wicket*, https://wicket.apache.org, accessed: 2021-02-01
[19] A. Corso Radu, G. Lehmann Miotto, L. Magnoni, J. Phys. Conf. Ser. **396**, 012014 (2012)
[20] A. Corso-Radu, L. Magnoni, R.M. Garcia, *The ELisA facility - RESTful API and client libraries*, in *2013 IEEE Nuclear Science Symposium and Medical Imaging Conference (2013 NSS/MIC)* (2013), pp. 1–4
[21] A. Corso-Radu, G. Avolio, Journal of Physics: Conference Series **1525**, 012029 (2020)